;-------------------------------------------------------------------------

;

; The WOZ Monitor for the Apple 1

; Written by Steve Wozniak 1976

;

; Converted to assemble with ca65 by cbmeeks

;

; Original code with comments taken from

; http://www.sbprojects.com/projects/apple1/wozmon.php

;

;-------------------------------------------------------------------------

.debuginfo +

.setcpu "65C02"

.ORG $FF00

;-------------------------------------------------------------------------

; Memory declaration

;-------------------------------------------------------------------------

XAML := $24 ; Last "opened" location Low

XAMH := $25 ; Last "opened" location High

STL := $26 ; Store address Low

STH := $27 ; Store address High

L := $28 ; Hex value parsing Low

H := $29 ; Hex value parsing High

YSAV := $2A ; Used to see if hex value is given

MODE := $2B ; $00=XAM, $7F=STOR, $AE=BLOCK XAM

IN := $0200 ; Input buffer ($0200 - $027F)

KBD := $D010 ; PIA.A keyboard input

KBDCR := $D011 ; PIA.A keyboard control register

DSP := $D012 ; PIA.B display output register

DSPCR := $D013 ; PIA.B display control register

; KBD b7..b0 are inputs, b6..b0 is ASCII input, b7 is constant high

; Programmed to respond to low to high KBD strobe

; DSP b6..b0 are outputs, b7 is input

; CB2 goes low when data is written, returns high when CB1 goes high

; Interrupts are enabled, though not used. KBD can be jumpered to IRQ,

; whereas DSP can be jumpered to NMI.

;-------------------------------------------------------------------------

; Constants

;-------------------------------------------------------------------------

BS := $DF ; Backspace key, arrow left key

CR := $8D ; Carriage Return

ESC := $9B ; ESC key

PROMPT := '\' ; Prompt character

;-------------------------------------------------------------------------

; Let's get started

;

; The RESET routine is only to be entered by asserting the RESET line of

; the system. This ensures that the data direction registers are selected.

;-------------------------------------------------------------------------

**RESET:**

CLD ; Clear decimal arithmetic mode

CLI

LDY #%01111111 ; Mask for DSP data direction reg

STY DSP ; (DDR mode is assumed after reset)

LDA #%10100111 ; KBD and DSP control register mask

STA KBDCR ; Enable interrupts, set CA1, CB1 for

STA DSPCR ; positive edge sense/output mode.

; Program falls through to the GETLINE routine to save some program bytes

; Please note that Y still holds $7F, which will cause an automatic Escape

;-------------------------------------------------------------------------

; The GETLINE process

;-------------------------------------------------------------------------

**NOTCR:**

CMP #BS ; Backspace key?

BEQ BACKSPACE ; Yes

CMP #ESC ; ESC?

BEQ ESCAPE ; Yes

INY ; Advance text index

BPL NEXTCHAR ; Auto ESC if line longer than 127

**ESCAPE:**

LDA #PROMPT ; Print prompt character

JSR ECHO ; Output it

**GETLINE:**

LDA #CR ; Send CR

JSR ECHO

LDY #0+1 ; Start a new input line

**BACKSPACE:**

DEY ; Backup text index

BMI GETLINE ; Oops, line empty, reinitialize

**NEXTCHAR:**

LDA KBDCR ; Wait for key press

BPL NEXTCHAR ; No key yet!

LDA KBD ; Load character. B7 should be '1'

STA IN,Y ; Add to text buffer

JSR ECHO ; Display character

CMP #CR

BNE NOTCR ; not CR!

; Line received, now let's parse it

LDY #-1 + 255 ; Reset text index

LDA #0 ; Default mode is XAM

TAX ; X=0

**SETSTOR:**

ASL ; Leaves $7B if setting STOR mode

**SETMODE:**

STA MODE ; Set mode flags

**BLSKIP:**

INY ; Advance text index

**NEXTITEM:**

LDA IN, Y ; Get character

CMP #CR

BEQ GETLINE ; We're done if it's CR

CMP #'.'

BCC BLSKIP ; Ignore everything below "."

BEQ SETMODE ; Set BLOCK XAM mode ("." = $AE)

CMP #':'

BEQ SETSTOR ; Set STOR mode. $BA will become $7B

CMP #'R'

BEQ RUN ; Run the program. Forget the rest.

STX L ; Clear input value (X=0)

STX H

STY YSAV ; Save Y for comparison

; Here we're trying to parse a new hex value

**NEXTHEX:**

LDA IN,Y ; Get character for hex test

EOR #$B0 ; Map digits to 0-9

CMP #9+1 ; Is it a decimal digit?

BCC DIG ; Yes!

ADC #$88 ; Map letter "A"-"F" to $FA-FF

CMP #$FA ; Hex letter?

BCC NOTHEX ; No! Character not hex

**DIG:**

ASL

ASL ; Hex digit to MSD of A

ASL

ASL

LDX #4 ; Shift count

**HEXSHIFT:**

ASL ; Hex digit left, MSB to carry

ROL L ; Rotate into LSD

ROL H ; Rotate into MSD

DEX ; Done 4 shifts?

BNE HEXSHIFT ; No, loop

INY ; Advance text index

BNE NEXTHEX ; Always taken

**NOTHEX:**

CPY YSAV ; Was at least 1 hex digit given?

BEQ ESCAPE ; No! Ignore all, start from scratch

BIT MODE ; Test MODE byte

BVC NOTSTOR ; B6=0 is STOR, 1 is XAM or BLOCK XAM

; STOR mode, save LSD of new hex byte

LDA L ; LSDs of hex data

STA (STL,X) ; Store current 'store index'(X=0)

INC STL ; Increment store index.

BNE NEXTITEM ; No carry!

INC STH ; Add carry to 'store index' high

**TONEXTITEM:**

JMP NEXTITEM ; Get next command item.

;-------------------------------------------------------------------------

; RUN user's program from last opened location

;-------------------------------------------------------------------------

**RUN:**

JMP (XAML) ; Run user program

;-------------------------------------------------------------------------

; We're not in Store mode

;-------------------------------------------------------------------------

**NOTSTOR:**

BMI XAMNEXT ; B7 = 0 for XAM, 1 for BLOCK XAM

; We're in XAM mode now

LDX #2 ; Copy 2 bytes

**SETADR:**

LDA L-1, X ; Copy hex data to

STA STL-1, X ; 'store index'

STA XAML-1, X ; and to 'XAM index'

DEX ; Next of 2 bytes

BNE SETADR ; Loop unless X = 0

; Print address and data from this address, fall through next BNE.

**NXTPRNT:**

BNE PRDATA ; NE means no address to print

LDA #CR ; Print CR first

JSR ECHO

LDA XAMH ; Output high-order byte of address

JSR PRBYTE

LDA XAML ; Output low-order byte of address

JSR PRBYTE

LDA #':' ; Print colon

JSR ECHO

**PRDATA:**

LDA #' ' ; Print space

JSR ECHO

LDA (XAML,X) ; Get data from address (X=0)

JSR PRBYTE ; Output it in hex format

**XAMNEXT:**

STX MODE ; 0 -> MODE (XAM mode).

LDA XAML ; See if there is more to print

CMP L

LDA XAMH

SBC H

BCS TONEXTITEM ; Not less! No more data to output

INC XAML ; Increment 'examine index'

BNE MOD8CHK ; No carry!

INC XAMH

**MOD8CHK:**

LDA XAML ; If address MOD 8 = 0 start new line

AND #%00000111

BPL NXTPRNT ; Always taken.

;-------------------------------------------------------------------------

; Subroutine to print a byte in A in hex form (destructive)

;-------------------------------------------------------------------------

**PRBYTE:**

PHA ; Save A for LSD

LSR

LSR

LSR ; MSD to LSD position

LSR

JSR PRHEX ; Output hex digit

PLA ; Restore A

; Fall through to print hex routine

;-------------------------------------------------------------------------

; Subroutine to print a hexadecimal digit

;-------------------------------------------------------------------------

**PRHEX:**

AND #%00001111 ; Mask LSD for hex print

ORA #'0' ; Add "0"

CMP #'9'+1 ; Is it a decimal digit?

BCC ECHO ; Yes! output it

ADC #6 ; Add offset for letter A-F

; Fall through to print routine

;-------------------------------------------------------------------------

; Subroutine to print a character to the terminal

;-------------------------------------------------------------------------

**ECHO:**

BIT DSP ; DA bit (B7) cleared yet?

BMI ECHO ; No! Wait for display ready

STA DSP ; Output character. Sets DA

RTS

;-------------------------------------------------------------------------

; Vector area

;-------------------------------------------------------------------------

.word $0000 ; Unused, what a pity

NMI\_VEC: .word $0F00 ; NMI vector

RESET\_VEC: .word RESET ; RESET vector

IRQ\_VEC: .word $0000 ; IRQ vector

;-------------------------------------------------------------------------